BAB IV
ANALISIS KEBUTUHAN DAN PERANCANGAN

Pada bab ini dibahas mengenai analisis kebutuhan dan perancangan terhadap sistem yang dibangun. Dilakukan proses analisis kebutuhan yang ada untuk selanjutnya dibentuk dalam suatu perancangan sistem yang meliputi perancangan sistem secara umum serta perancangan secara khusus.

4.1 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari aplikasi yang akan dibangun. Kebutuhan dari aplikasi yang dibangun meliputi kebutuhan dari perangkat keras, perangkat lunak dan data citra telapak tangan.

Aplikasi yang akan dibuat adalah aplikasi yang akan mengenali ujung jemari dan jumlah jari yang direntangkan dari telapak tangan, sehingga dari isyarat jari yang telah ditentukan sebelumnya akan diproses lebih lanjut sehingga dapat digunakan dalam menggerakkan *mouse pointer* komputer. Metode yang digunakan dalam aplikasi ini untuk mendapatkan ujung dari jemari adalah metode pendeteksian *corner* menggunakan algoritma *k-curvature* atau *k-cosine.* Secara umum aplikasi membutuhkan sebuah perangkat komputer disertai dengan kamera dan perangkat lunak untuk menampilkan hasil proses ekstraksi telapak tangan. Secara umum kebutuhan dari sistem dapat dideskripsikan sebagai berikut:

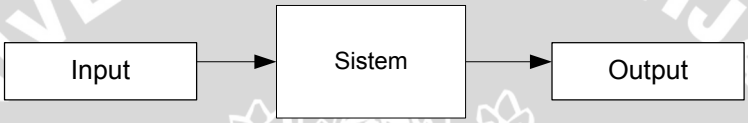
a. Pengguna dapat mengoperasikan *mouse pointer* komputer.
b. Pengguna dapat melihat hasil *preprocessing* dan ekstraksi ciri citra.
c. Aplikasi dapat menampilkan hasil *error rate* dan *mean square error* dari isyarat yang diberikan.
d. Aplikasi dapat menerima masukan citra dari *webcam.*
e. Aplikasi dapat mengetahui jumah jari yang direntangkan.
f. Aplikasi dapat menerjemahkan isyarat tangan yang diberikan menjadi *event mouse pointer* dengan benar.
g. Pengguna dapat mengontrol untuk mengaktifkan atau menonaktifkan operasi menggerakkan *mouse pointer.*

4.2 Perancangan Perangkat Lunak

Aplikasi yang dirancang hanya digunakan sebagai alat bantu dalam mengekstraksi ciri telapak tangan dengan menggunakan algoritma K-curvature. Perancangan aplikasi dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisis kebutuhan. Dari analisis, kita dapat menentukan rancangan masukan dan keluaran aplikasi, rancangan tahapan proses dan rancangan tampilan aplikasi, *user interface*.

4.2.1 Diagram Blok

Gambar di bawah ini menggambarkan diagram blok dari sistem yang dibangun yang meliputi *input* serta *output* dari sistem.



Gambar 4.1 Diagram Blok Sistem
Sumber : Perancangan

a. Masukan Aplikasi

Aplikasi yang akan dirancang diharapkan dapat mengekstraksi ciri citra telapak tangan dengan tingkat kesalahan seminimal mungkin. Aplikasi yang akan dirancang ini akan memproses citra gambar yang di dalamnya terdapat objek telapak tangan. Citra digital tersebut diperoleh dari hasil tangkapan kamera lokal komputer. Citra yang ditangkap berupa citra dengan ukuran 160 x 120 pixel dengan format RGB. Dikarenakan penelitian ini dikhususkan dalam ekstraksi ciri telapak tangan saja, maka tangkapan kamera akan di-crop untuk kemudian diambil daerah gambar di pojok kanan atas saja. Hal ini dimaksudkan untuk mengurangi kemungkinan tertangkapnya objek lain selain telapak tangan oleh kamera.

b. Keluaran Aplikasi

Aplikasi yang dirancang, akan menerapkan algoritma *K-curvature* atau juga dapat disebut dengan *K-cosine*. Algoritma tersebut diharapkan dapat mendeteksi lengkungan-lengkungan dari objek dalam citra. Lengkungan tersebut merupakan representasi dari ujung jari pada telapak tangan. Aplikasi juga akan menampilkan citra hasil proses ekstraksi. Hasil citra berupa objek akan berwarna

putih dan latar belakang akan berwarna hitam. Untuk mempermudah kita mengetahui daerah mana saja yang berhasil terdeteksi sebagai ujung jari maka daerah tersebut oleh aplikasi akan diberikan warna kuning. Selain hasil ekstraksi dari citra, keluaran dari aplikasi juga berupa *event mouse pointer*. *Event mouse* ini digunakan untuk pengoperasian *mouse pointer* sehingga pengguna dapat menggerakkan *mouse pointer.*

4.2.2 Diagram Kelas

Aplikasi yang akan dibangun menggunakan lingkup bahasa pemrogaman java, sehingga aplikasi ini tersusun dari beberapa kelas. Kelas yang dibutuhkan antara lain :

1. Kelas Main

Kelas Main merupakan kelas utama atau kelas yang dijalankan pertama kali. Kelas main akan memanggil kelas-kelas lainya yang digunakan untuk membantu *preprocessing* dan menangkap citra dari *webcam*. Di dalam kelas ini juga dideklarasiakn kelas-kelas yang mendukung proses mendeteksi jumlah ujung jari yang direntangkan. Kelas Main adalah hasil kelas turunan dari kelas Capture. Pada konstruktor kelas main dideklarasikan tentang elemen-elemen interface dari tampilan aplikasi, selain itu juga dideklarasikan fungsi-fungsi untuk menangkap citra webcam dan disimpan di variabel sementara.

2. Kelas Capture

Merupakan kelas abstrak yang di dalamnya dideklarasikan beberapa konfigurasi dari aplikasi. Konfigurasi ini dimaksudkan untuk menghubungkan aplikasi dengan kamera lokal komputer sehingga nantinya aplikasi dapat menangkap citra dari kamera. Konfigurasi tersebut adalah mendeklarasikan *path* letak *library* VLC yang telah tersintall di sistem operasi.

3. Kelas Binary Region

Kelas yang nantinya digunakan sebagai tempat penyimpanan pixel-pixel yang tersusun dalam kesatuan daerah atau *region.* Di dalam kelas ini terdapat berbagai fungsi-fungsi diantaranya adalah getLabel, getSize, getBoundingBox, getCenter, addPixel, update, getXp, getYp, manhattan dan toString. Fungsi getLabel digunakan untuk mendapatkan nama label dari region yang ditunjuk. Fungsi getSize digunakan untuk mendapatkan ukuran dari region yang ditunjuk.

Fungsi getBoundingBox mengembalikan sebuah daerah persegi empat dari daerah terluar dari region. Fungsi getCenter mengembalikan kordinat pusat dari region. Fungsi addPixel digunakan untuk menambahkan pixel untuk region. Fungsi update digunakan untuk memperbarui nilai-nilai atribut. Fungsi getXp mengembalikan kordinat x pusat berdasarkan jarak manhattan. Fungsi getYp mengembalikan kordinat y pusat berdasarkan jarak manhattan. Fungsi manhattan digunakan untuk menghitung jarak manhatan dari region. Fungsi toString digunakan untuk menampilkan keterangan dari region.

4. Kelas Contour

Kelas yang nantinya digunakan sebagai tempat penyimpanan kontour yang berhasil dideteksi oleh aplikasi. Di dalam kelas Contour terdapat fungsi-fungsi diantaranya adalah addPoint, getPoint, makePolygon, getLength, toString. Fungsi addPoint digunakan untuk menambahkan titik dari kontor yang telah ditelusuri. Fungsi getPoint digunakan untuk mengembalikan titik kontur. Fungsi makePolygon mengembalikan shape dari kontur yang telah didapatkan. Fungsi getLength mengembalikan banyaknya titik pada kontur. Fungsi toString digunakan untuk menampilkan keterangan dari kontur.

5. Kelas ContourTracer

Merupakan kelas yang digunakan untuk medeteksi atau mendapatkan kontour dari citra biner. Kelas ini akan menelusuri objek yang terdeteksi dan mendapatkan kontur dari objek tersebut.

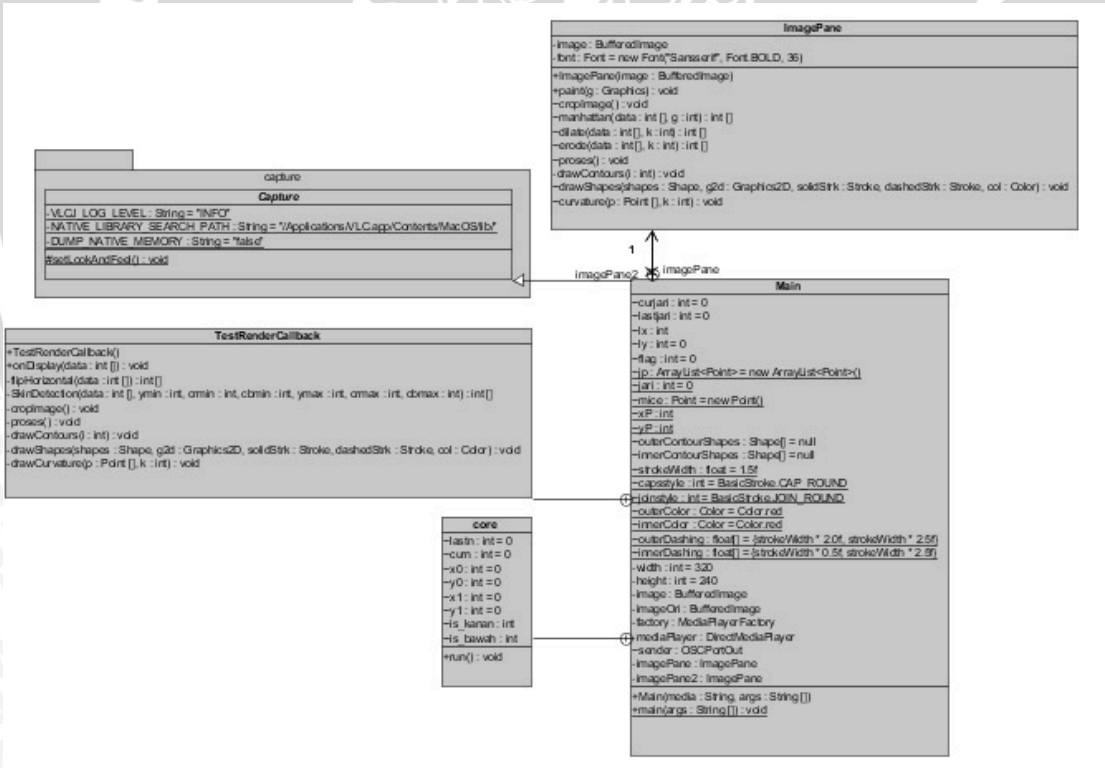
6. Kelas TestRenderCallback

Kelas yang berada di dalam kelas main, kelas utama. Selain tempat pendeklarasian fungsi pengolahan citra digital, kelas ini juga difungsikan untuk menangkap citra dari kamera dan disimpan dalam variabel image. Dari data yang tersimpan tadi, dapat diproses lebih lanjut atau langsung ditampilkan pada form aplikasi.

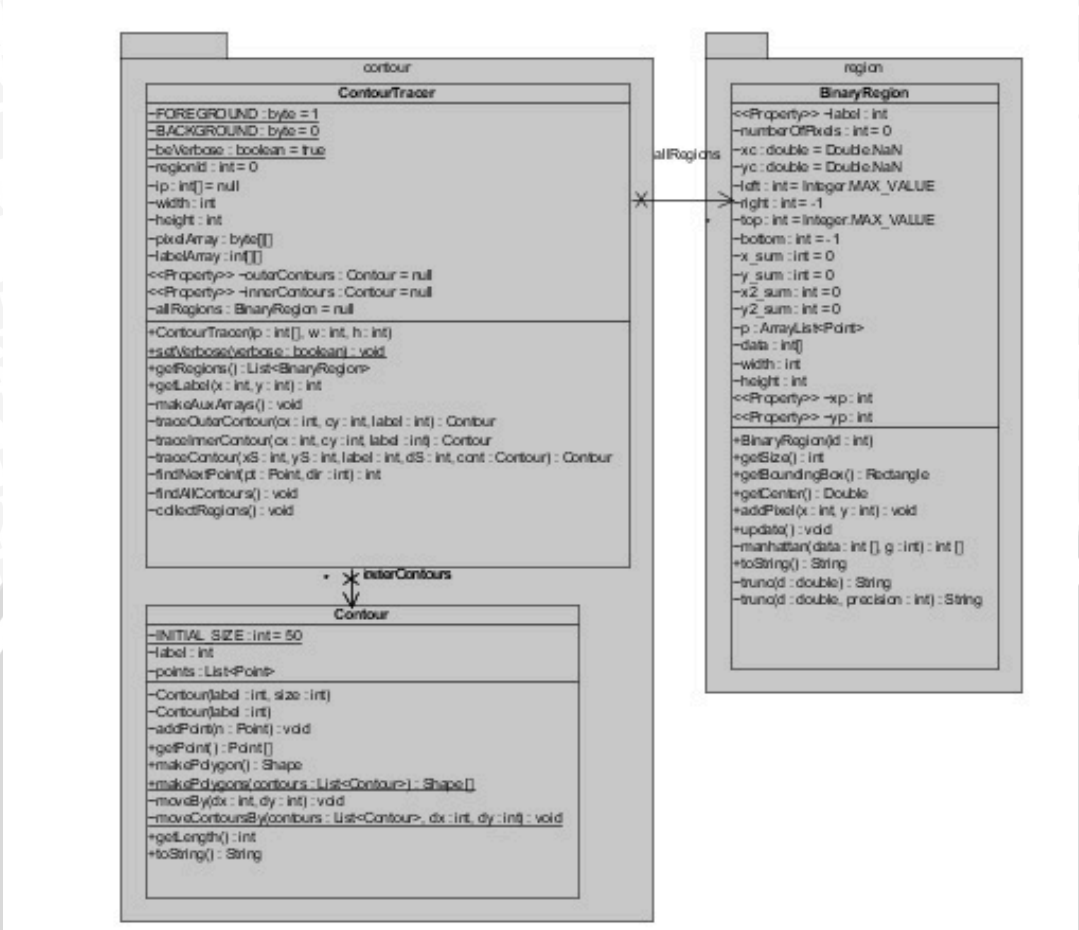
Terdapat fungsi-fungsi utama di dalam kelas TestRenderCallback yaitu: onDisplay, flipHorizontal, skinDetection, cropImage, proses, drawContour, drawShape, drawCurvature. Fungsi onDisplay digunakan untuk menampilkan citra hasil tangkapan webcam maupun citra hasil ekstraksi ke jendela aplikasi. Fungsi flipHorizontal digunakan untuk membalik citra secara Horizontal. Fungsi

skinDetection digunakan untuk memilih daerah mana saja yang warnanya termasuk dalam warna yang menyerupai warna kulit. Untuk warna yang menyerupai warna kulit diberi warna putih dan hitam untuk warna selain warna kulit. Fungsi cropImage digunakan untuk mengambil bagian tertentu saja dari gambar. Fungsi drawContur digunakan untuk mendapatkan kontur objek dan menandai daerah kontur yang telah ditelusuri. Di dalam fungsi drawContur juga dipanggil fungsi drawShape. Fungsi proses digunakan untuk mendeteksi sudut kelengkungan dari kontur.

Dari deskripsi kelas di atas, terdapat banyak sekali hubungan antara kelas satu dengan kelas yang lain. Untuk mempermudah dalam merepresentasikan hubungan antara kelas-kelas tersebut, berikut ini dicantumkan diagram kelas dari aplikasi yang akan dibangun.

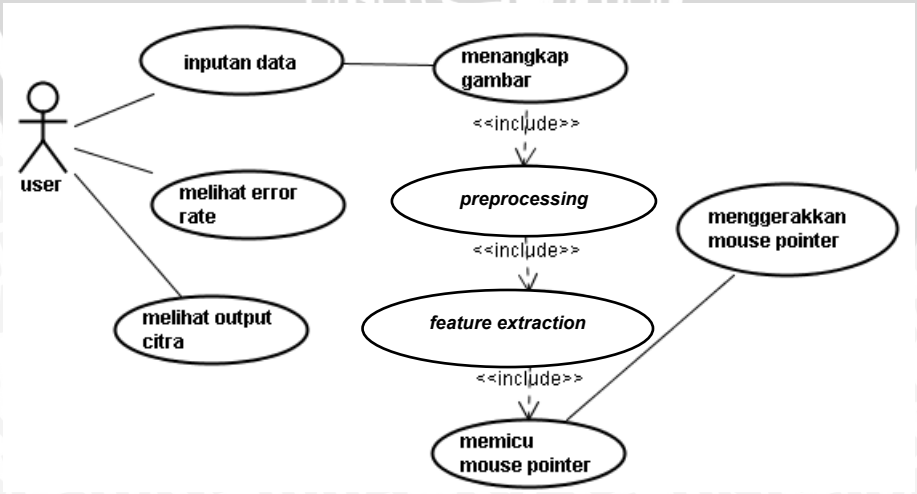


(lanjut ke halaman berikutnya)



Gambar 4.2 Diagram Kelas Aplikasi
Sumber : Perancangan

4.2.3 Diagram *Use Case*



Gambar 4.3 Diagram *Use Case* Aplikasi
Sumber : Perancangan

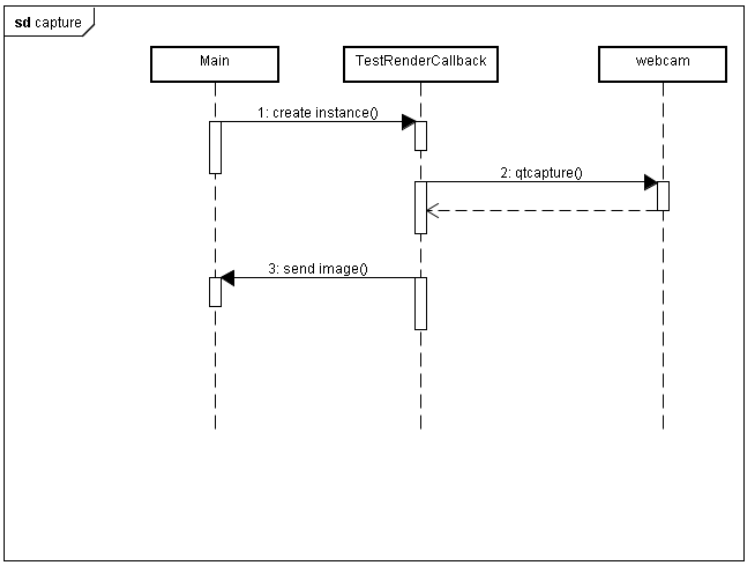
Gambar di atas, menggambarkan diagram *use case* dari sistem yang dibangun. Terdapat beberapa *case* untuk memenuhi kebutuhan pengguna. Pada aplikasi yang dirancang, didefinisikan dua buah aktor yaitu pengguna dan sistem operasi. Sistem yang sedang dibuat merupakan perantara pengguna dengan sistem operasi sehingga isyarat tangan yang diberikan oleh pengguna dapat diterjemahkan dengan benar menjadi *event mouse pointer* ke sistem operasi. Untuk melakukan proses tersebut, di dalam aplikasi mempunyai beberapa tahap yaitu menangkap gambar, *preprocessing* citra, mendeteksi jumlah jari / *feature extraction,* memicu *event mouse pointer*. Aplikasi akan menangkap objek tangan yang diterima dari kamera lokal komputer untuk kemudian dilakukan *preprocessing* pada citra. Pada tahap *preprocessing* ini pengguna dapat langsung mengetahui hasil *preprocessing* pada citra, dimaksudkan agar pengguna selaku penulis dapat mengetahui apabila terjadi kesalahan ketika *preprocessing.* Tahap deteksi jumlah jari yaitu ketika aplikasi mengekstraksi sudut kelengkungan kontur telapak tangan dan mencari kelengkungan yang termasuk ujung jari saja. Pada tahap ekstraksi ciri ini pengguna dapat langsung mengetahui tingkat kesalahan dari hasil deteksi jumlah jari yang direntangkan. Pada tahap memicu / pengoperasian *event mouse pointer* aplikasi akan menerjemahkan hasil ekstraksi sebelumnya ke *events* dari *mouse pointer*. Pengguna dapat mengaktifkan atau menonaktifkan proses pengoperasian *mouse pointer*. Pada tahap pengoperasian ini, aplikasi akan mengirim pesan ke sistem operasi terkait dengan *event mouse pointer* yang ingin dilakukan.

4.2.4 Diagram *Sequence*

Pada diagram sequence akan dijelaskan secara lebih khusus mengenai hubungan proses satu dengan proses lainya yang dikerjakan oleh aplikasi. Proses-proses yang dikerjakan oleh tiap kelas akan diatur hubunganya sesuai dengan urutan waktu.

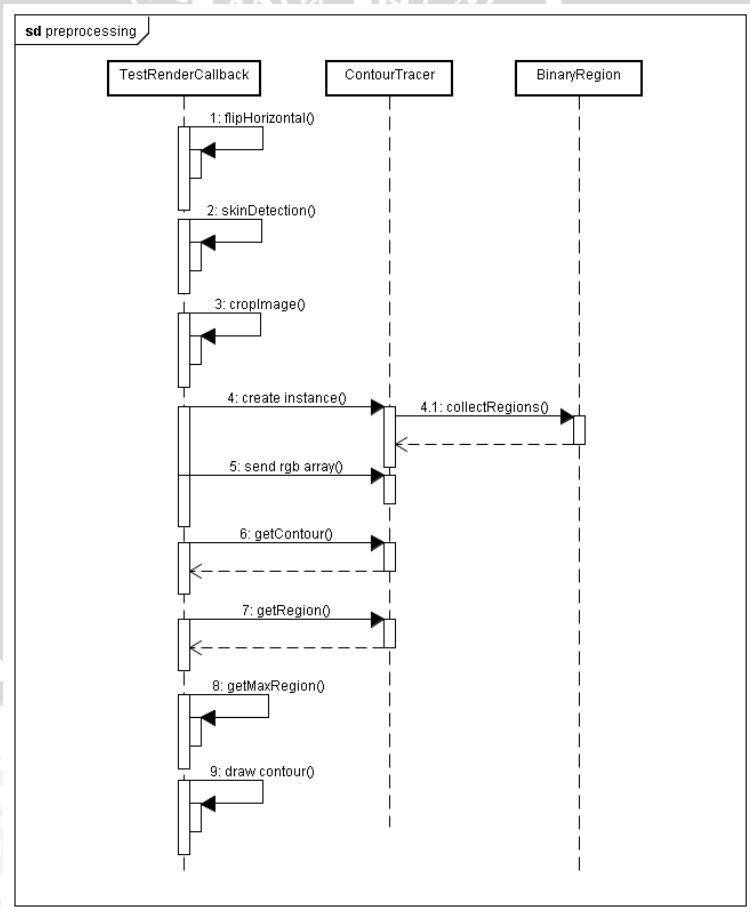
a. Menangkap Gambar

   Berikut adalah *sequence diagram* dari proses manangkap gambar menggunakan webcam.



Gambar 4.4 *Sequence Diagram* menangkap gambar
Sumber : Perancangan

b. Preprocessing



Gambar 4.5 *Sequence Diagram* proses preprocessing
Sumber : Perancangan

Untuk mendapatkan hasil segmentasi mendekati kenyataanya, maka pada proses ini citra yang semula berformat RGB dari kamera diubah menjadi YCbCr. Kemudian dengan batas range tertentu, yaitu terdapat batas nilai maksimal dan batas nilai minimal untuk tiap elemen warna Y, Cb dan Cr,dapat ditentukan warna yang mendekati warna kulit pada citra. Hasil akhir dari proses ini adalah citra biner dengan putih adalah objek tangan dan warna hitam adalah latar citra.

Untuk mendapatkan hasil segmentasi mendekati kenyataanya, maka pada proses ini citra yang semula berformat RGB dari kamera diubah menjadi YCbCr. Kemudian dengan batas range tertentu, yaitu terdapat batas nilai maksimal dan batas nilai minimal untuk tiap elemen warna Y, Cb dan Cr,dapat ditentukan warna yang mendekati warna kulit pada citra. Berbeda dengan RGB, ruang warna YCbCr adalah luma-independen, menghasilkan kinerja yang lebih baik. Cluster kulit yang bersangkutan akan diberikan sebagai [KUK-2004]:

$Y>80$

$85<Cb<135$

$135<Cr<180$

Dimana Y,Cb,Cr = [0,255]

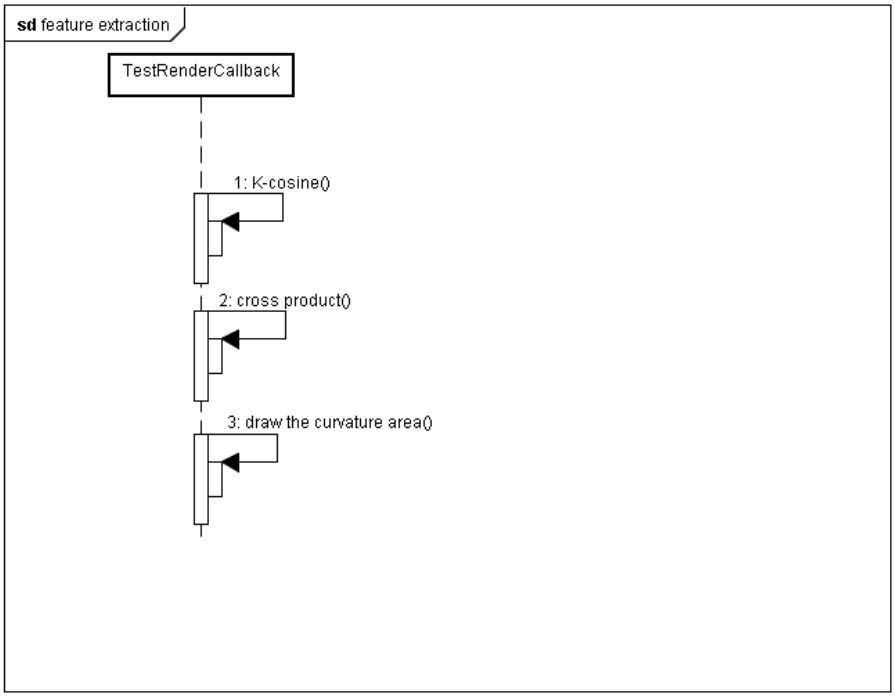
Hasil akhir dari proses ini adalah citra biner dengan putih adalah objek tangan dan warna hitam adalah latar citra. Selanjutnya citra biner diberi label atau diberi indeks unik. Setiap objek yang telah dilabel tadi akan dicari konturnya.

c. Ekstraksi Fitur

Tiap pixel atau titik di sepanjang garis kontur dari objek, akan ditelusuri untuk mendapatkan titik kordinat dengan sudut lengkungan yang diinginkan. Algoritma *k-cosine* digunkan untuk mendapatkan sudut kelengkungan dari titik kordinat yang ditelusuri. Rentangan nilai yang dihasilkan antara 1 sampai dengan -1 atau setara dengan 0 sampai 180 derajat.

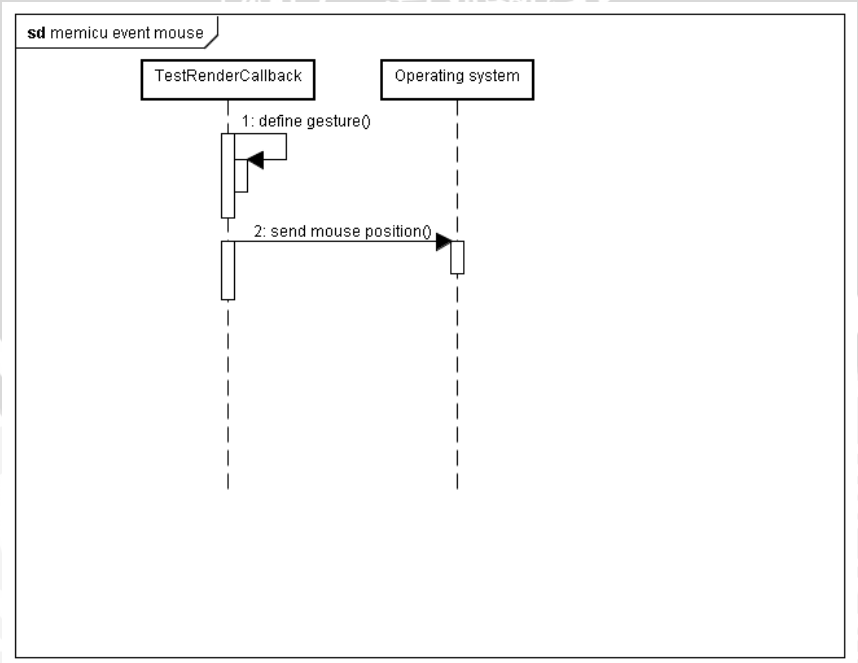
Aplikasi akan mencari lengkungan pada bagian ujung jemari berdasarkan sudut *threshold* kelengkungan. Dari kordinat yang telah didapatkan sebelumnya, dapat dibuat vektor untuk sumbu x, y dan z. Perkalian vektor digunakan untuk mendapatkan nilai z sebagai pembeda antara lengkungan ujung jari dan lengkungan sela-sela jari. Untuk nilai z yang kurang dari 0 didefinisikan sebagai

lengkungan ujung jari dan z lebih dari sama dengan 0 didefinisikan sebagai lengkungan sela-sela jari.



Gambar 4.6 *Sequence Diagram* Ekstraksi Fitur
Sumber : Perancangan

d. Memicu *Mouse Pointer*



Gambar 4.7 *Sequence Diagram* Memicu Mouse Pointer
Sumber : Perancangan

Dari jumlah jari yang berhasil didapatkan, aplikasi akan menerjemahkan kombinasi dari jumlah jari yang direntangkan ke dalam bentuk-bentuk *event mouse pointer*. Berikut adalah penjelasan dari kombinasi jumlah jari yang direntangkan dengan *event mouse*.

Tabel 4.1 Rancangan Isyarat Tangan untuk Mengoperasikan *Mouse*

Kombinasi Jumlah Jari	Keterangan
Jumlah jari sama dengan 0	Diam
Jumlah jari sama dengan 1 atau 2	Menggerakkan mouse pointer
Jumlah jari dari 1 ke 2	Memicu untuk menekan tombol kiri *mouse*
Jumlah jari dari 2 ke 1	Memicu untuk melepas tombol kiri *mouse*
Jumlah jari dari 0 ke 5 atau sebaliknya	Membuka *lauchpad*
Jumlah jari dari 4	Ganti ke *desktop* lainya
Jumlah jari dari 3 dan bergerak ke atas	Memicu untuk menekan tombol panah ke atas
Jumlah jari dari 3 dan bergerak ke bawah	Memicu untuk menekan tombol panah ke bawah
Jumlah jari dari 3 dan bergerak ke kanan	Memicu untuk menekan tombol panah ke kanan
Jumlah jari dari 3 dan bergerak ke kiri	Memicu untuk menekan tombol panah ke kiri
Jumlah jari dari 1 ke 5	Memicu untuk klik kanan tombol *mouse*

Setelah daerah lengkungan ditemukan, dari daerah tersebut dihitung koordinatnya yang nantinya digunakan sebagai representasi koordinat *mouse pointer* pada layar.

4.2.5 Perancangan Algoritma

Perancangan algoritma berupa *pseudocode* yang merupakan perancangan algoritma dalam bahasa alami yang kemudian dituangkan dalam bahasa

pemrograman. Untuk perancangan algoritma juga dapat direpresentasikan dalam bentuk diagram alir.

Berikut adalah algoritma yang dirancang untuk mendeteksi lengkungan ujung jari:

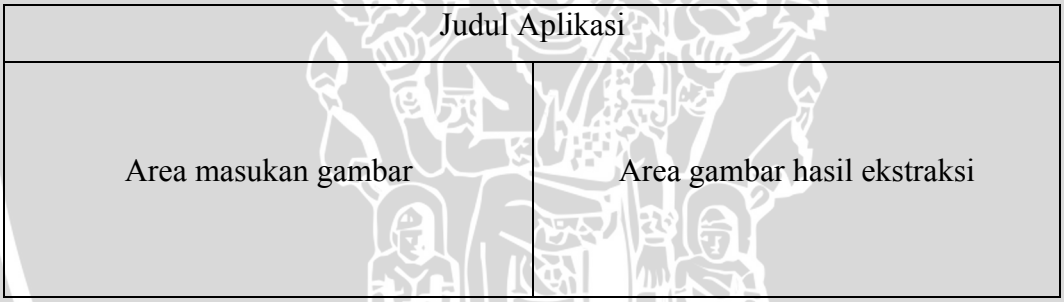
```
    Nama algoritma :
    K-cosine Corner Detection  + Cross Product
    Deklarasi :
       a. Integer K,theta -> variable konstanta K, dan sudut
          threshold.
       b. Array p -> menampung kumpulan pixel dari kontur
       c. Pleft -> menampung kordinat titik di sebelah kiri
       d. Pright -> menampung kordinat titik di sebelah
          kanan
       e. Lb -> menampung jarak titik ke i ke titik Pright
       f. Lf -> menampung jarak titik ke i ke titik Pleft
       g. xa -> menampung niali x vector a
       h. ya -> menampung nilai y vector a
       i. xb -> menampung niali x vector b
       j. yb -> menampung nilai y vector b
       k. x2 -> menampung nilai sudut kelengkungan

    Proses Algoritma :
1   INPUT p, K, theta
2
3   FOR p indeks i ke 0 sampai indeks terakhir
4    Masukkan titik K pixel sebelah kiri ke Pleft
5    Masukkan titik K pixel sebelah kanan ke Prigth
6    Cari jarak Euclidean Pright ke p[i] dan masukkan ke lb
     Cari jarak Euclidean Pleft ke p[i] dan masukkan ke lf
7    Dari titk p[i] ke titik Pleft (vector a) dan titik p[i]
     ke titik Pright (vector b)
8    Masukkan nilai x dan y untuk tiap vector a dan b pada
9    variabel xa,ya,xb,yb

10   Hitung menggunakan persamaan K-cosine (2-41) dan
11   masukkan ke variabel x2
12
13   IF x2 < theta
14      IF hasil cross product vector a dan b <0
15      THEN flaging p[i] sebagai ujung jari
16      IF hasil cross product vector a dan b >=0
17      THEN flaging p[i]
18   ENDIF
19  ENDFOR
```

Gambar 4.8 *Pseudocode*
Sumber : Perancangan

Baris 2 sampai dengan baris 9 adalah penggunaan persamaan K-cosine (2-14) untuk mendapatkan kelengkungan kontur telapak tangan. Dari kelengkungan yang telah didapatkan akan dibedakan kelengkungan ujung jari dan kelengkungan antara kedua jari. Untuk membedakanya aplikasi menggunakan persamaan *cross product* (2-43) dari kedua vektor a dan b seperti pada baris ke 12 dan 14. Untuk nilai z kurang dari 0 merupakan daerah yang terdeteksi sebagai ujung jari.

4.2.6 Perancangan Tampilan Aplikasi

Tampilan utama dari aplikasi yang mengekstraksi ciri citra telapak tangan sebagai pengoperasian pointer *mouse* komputer dapat dilihat pada Gambar 4.4. Aplikasi menjalankan proses ekstraksi pada gambar atau citra masukan yang berasal dari kamera. Citra tersebut nantinya akan ditampilkan pada form aplikasi. Terdapat dua buah citra yang ditampilkan pada form, yaitu citra asli yang ditangkap kamera, berada di sebelah kiri dan citra hasil proses pengolahan citra digital, berada di sebelah kanan.



Gambar 4.9 Rancangan tampilan form aplikasi
Sumber : Perancangan

4.2.6 Perancangan Pengujian

Pada skripsi ini penulis berusaha melakukan beberapa jenis pengujian algoritma dan aplikasi secara umum. Pengujian algoritma dilakukan dengan mencatat rata-rata waktu selama pemrosesan dan mengevaluasi tingkat keberhasilan algoritma K-cosine dalam mendeteksi jumlah jari terhadap perubahan nilai variabel uji. Beberapa variabel uji dalam skripsi ini adalah sebagai berikut: nilai *k* dari algoritma K-cosine, nilai *threshold* dari sudut kelengkungan dan ukuran panjang, lebar citra. Pengujian algoritma ini berfungsi untuk mendapatkan nilai *k* dan sudut kelengkungan yang optimal untuk tiap ukuran gambar yang berbeda. Dengan nilai yang optimal tersebut akan diperoleh hasil ekstraksi dengan tingkat kesalahan seminimal mungkin.

Pengujian aplikasi dilakukan dengan menggunakan masukan citra telapak tangan dengan kondisi warna kulit terang dan gelap untuk menggerakkan *mouse pointer* komputer. Untuk menggerakkan *mouse* pointer komputer, sebelumnya aplikasi harus dapat mengenali isyarat telapak tangan yang diberikan dengan benar. Mengenali isyarat tangan dapat dikatakan berhasil ketika aplikasi berhasil mendapatkan daerah tangan saja sesuai jangkauan warna yang telah ditentukan. Oleh karena itu, dilakukanlah uji coba pemrosesan citra telapak tangan dengan warna kulit terang dan gelap untuk mendapatkan tingkat keberhasilan dari aplikasi berdasarkan kondisi yang telah diberikan.

4.2.7 Skenario Pengujian

Pengujian dilakukan dengan membandingkan jumlah jari yang direntangkan oleh pengguna dengan jumlah jari yang berhasil dideteksi oleh aplikasi. Selain melakukan pengujian dengan perbandingan sederhana, pengujian juga dilakukan dengan menghitung MSE dari tiap perbedaan perlakuan ketika pengujian. Persamaan MSE yang digunakan dapat dilihat pada persamaan 2-43. Perhitungan kesalahan dilakukan dengan menguji 300 frame citra masukan. Tabel evaluasi berdasarkan sudut kelengkungan dan nilai *k* dapat dilihat pada tabel 4.2 dan tabel evaluasi berdasarkan nilai *k* dan ukuran citra dapat dilihat pada tabel 4.3. Tabel waktu pemrosesan dapat dilihat pada tabel 4.4. Untuk pengujian tabel 4.2 dilakukan dengan menggunakan ukuran citra 320x240 pixel. Pengujian tersebut dilakukan untuk mendapatkan sudut *threshold* optimal. Selanjutnya dengan sudut optimal yang diperoleh itu, sudut tersebut digunakan dalam pengujian tabel 4.3. Pengujian tersebut dilakukan untuk mencari keterkaitan atau hubungan antara perubahan ukuran citra dengan nilai *k*. Pengujian ini menggunakan tiga buah ukuran citra yang berbeda-beda. Setelah didapatkan sudut optimal dari tabel 4.1 peneliti akan mencari k optimal untuk ukuran citra 320x240. Hasil pengujian untuk mencari k optimal dapat dilihat pada tabel 4.5.

Tabel 4.2 Rancang tabel pengujian *k-cosine,* variabel *k* dan sudut *threshold*

k=32						
Jumlah Jari	Sudut 30		Sudut 60		Sudut 100	
	Kesalahan (%)	MSE	Kesalahan (%)	MSE	Kesalahan (%)	MSE
1						
2						
3						
4						
5						
Rata-rata						

k=64						
Jumlah Jari	Sudut 30		Sudut 60		Sudut 100	
	Kesalahan (%)	MSE	Kesalahan (%)	MSE	Kesalahan (%)	MSE
1						
2						
3						
4						
5						
Rata-rata						

Tabel 4.3 Rancang tabel pengujian *k-cosine,* variabel *k* dan ukuran citra

k=32						
Jumlah Jari	160x120		320x240		480x360	
	Kesalahan (%)	MSE	Kesalahan (%)	MSE	Kesalahan (%)	MSE
1						

2						
3						
4						
5						
Rata-rata						

k=64						
Jumlah Jari	160x120		320x240		480x360	
	Kesalahan (%)	MSE	Kesalahan (%)	MSE	Kesalahan (%)	MSE
1						
2						
3						
4						
5						
Rata-rata						

Tabel 4.4 Rancang tabel pengujian waktu pemprosesan

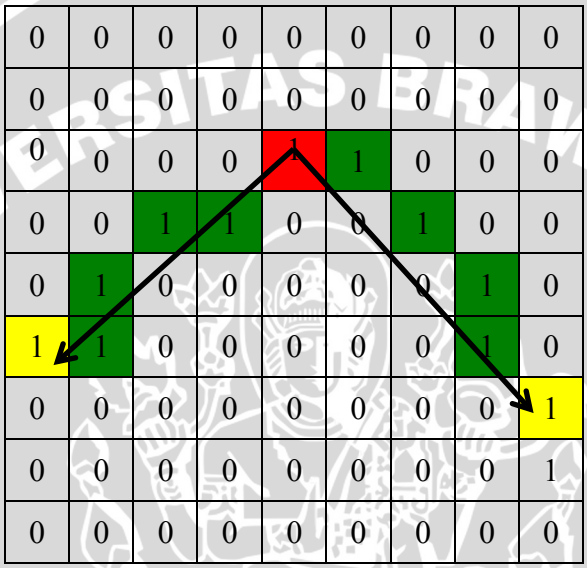
Ukuran Citra	Waktu	
	k=32	*k*=64
160 x 120		
320 x 240		
480 x 360		

Tabel 4.5 Rancangan tabel pengujian k optimal

Nilai K	Error Rate					MSE					
	1	2	3	4	Mean	1	2	3	4	5	Mean

4.3 Perhitungan Manual

Berikut ini akan dicontohkan proses perhitungan *K*-cosine dalam mendapatkan nilai kelengkungan dari garis kontur telapak tangan. Persamaan *K-cosine* dapat dilihat pada persamaan 2-41. Misalnya, penulis mendapatkan citra biner berupa garis kontur, garis kontur berwarna putih (bernilai 1) dan latar citra berwarna hitam (bernilai 0). Berikut ini adalah contoh lengkungan yang diambil dari salah satu bagian kontur citra.



Gambar 4.10 Citra Biner Garis Lengkung
Sumber : Perancangan

Pada contoh kasus ini, penulis akan mencari kelengkungan di titik yang ditandai dengan warna merah menggunakan algoritma *K*-cosine. Sebelumnya, harus didefinisikan terlebih dahulu nilai *K*-nya. Misalnya, nilai *K* yang dipakai adalah 5, maka titik tetangga ke-5 dari titik berwarna merah akan ditandai dengan warna kuning. Setelah ketiga koordinat atau letak titik diperoleh, titik kuning kiri dengan koordinat (0,3), titik merah dengan koordinat (4,6) dan titik kuning kanan dengan koordinat (8,2), maka dicari nilai matriks vektor dan jarak dari masing-masing vektor $\vec{a}$ dan vektor $\vec{b}$. Vektor $\vec{a}$ adalah vektor dari titik merah ke titik kuning sebelah kiri dan vektor $\vec{b}$ adalah vektor dari titik merah ke titik kuning sebelah kanan. Vektor didapatkan dari nilai koordinat titik kuning tetangga dikurangi dengan titik merah. Setelah dihitung maka nilai vektor $\vec{a} = \begin{bmatrix} -4 \\ -3 \end{bmatrix}$ dan

nilai vektor $\vec{b} = \begin{bmatrix} 4 \\ -4 \end{bmatrix}$. Dari rumus jarak *euclidean* diperoleh jarak masing-masing vektor adalah sebagai berikut:

$$\left|\left|\vec{a}\right|\right| = \sqrt{(-4)^2 + (-3)^2}$$
$$\left|\left|\vec{a}\right|\right| = 5$$

$$\left|\left|\vec{b}\right|\right| = \sqrt{(4)^2 + (-4)^2}$$
$$\left|\left|\vec{b}\right|\right| = \sqrt{32}$$

Setelah nilai vektor dan jarak diketahui maka nilai-nilai tersebut dimasukkan ke dalam persamaan *k*-cosine.

$$c_i(K) = \cos\theta_i = \frac{\overrightarrow{a_i}(K).\overrightarrow{b_i}(K)}{\left|\left|\overrightarrow{a_i}(K)\right|\right|.\left|\left|\overrightarrow{b_i}(K)\right|\right|}$$

$$c(5) = \cos\theta = \frac{\vec{a}(5).\vec{b}(5)}{\left|\left|\vec{a}(5)\right|\right|.\left|\left|\vec{b}(5)\right|\right|}$$

$$c(5) = \cos\theta = \frac{\begin{bmatrix} -4 \\ -3 \end{bmatrix}.\begin{bmatrix} 4 \\ -4 \end{bmatrix}}{5\times\sqrt{32}}$$

$$c(5) = \cos\theta = \frac{(-4\times4) + (-3\times-4)}{28{,}2842}$$

$$c(5) = \cos\theta = \frac{-4}{28{,}2842}$$

$$c(5) = \cos\theta = -\,0{,}141422$$

$$\therefore \theta = 98{,}13^\circ$$

Dari persamaan *K-cosine* didapatkan sudut yang membetuk kedua vektor adalah sudut tumpul. Nilai sudut inilah yang nantinya digunakan untuk mendapatkan lengkungan dari kontur telapak tangan yang termasuk ujung jari atau bukan.